RAIRO-Theor. Inf. Appl. 48 (2014) 85–105
DOI: 10.1051/ita/2014002

Available online at:
www.rairo-ita.org

TRANSDUCING BY OBSERVING LENGTH-REDUCING AND PAINTER RULES

Norbert Hundeshagen[1] and Peter Leupold[2]

Abstract. The recently introduced model of transducing by observing is compared with traditional models for computing transductions on the one hand and the recently introduced restarting transducers on the other hand. Most noteworthy, transducing observer systems with length-reducing rules are almost equivalent to RRWW-transducers. With painter rules we obtain a larger class of relations that additionally includes nearly all rational relations.

Mathematics Subject Classification. 68Q68, 68Q42.

1. Different models of transductions

Transduction are most commonly defined by means of devices derived from finite automata and pushdown automata: finite state transducers are probably the best known example, pushdown transducers are another one. In both cases, the devices work in the same way as the variants that accept languages, but in every transition they can additionally make an output. The concatenation of these outputs is the product of the transduction. The pairs of input and output form the relation that is computed.

In recent years two new computational models for transductions have been introduced. On the one hand, there are *restarting transducers*. These are based on restarting automata [11], which were introduced to model the so-called "analysis by reduction". This is a linguistic technique used to analyze sentences of natural languages with free word order. Originally the restarting automaton is a language

Keywords and phrases. Restarting automata, computing by observing, transductions.

[1] Arbeitsgruppe Theoretische Informatik, Fachbereich Elektrotechnik/Informatik, Universität Kassel, Germany. hundeshagen@theory.informatik.uni-kassel.de

[2] Institut für Informatik, Universität Leipzig, Germany. Peter.Leupold@web.de

Article published by EDP Sciences

© EDP Sciences 2014

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

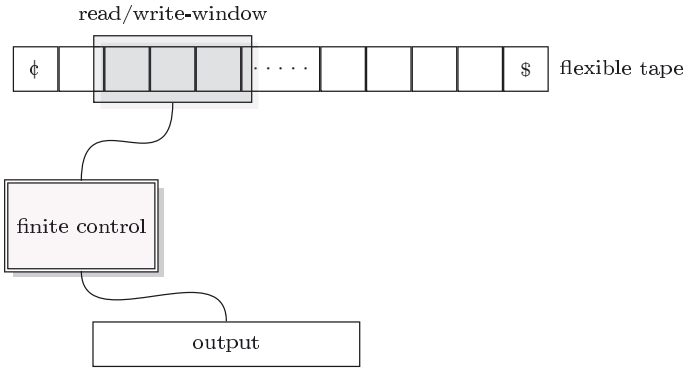


FIGURE 1. Schematic representation of a restarting transducer.

accepting device. The real goal of performing analysis by reduction, however, is not simply to accept or reject a given input sentence, but to extract information from that sentence and to translate it into another form, *e.g.*, into a formal representation. To this end restarting transducers were first introduced by Hundeshagen and Otto [10]. A restarting transducer consists of a finite-state control, a flexible tape with end markers, a read/write window of a fixed size working on that tape, and a write-only oneway output tape.

A schematic representation of a restarting transducer is shown in Figure 1. It works in cycles. In each cycle it performs a single rewrite operation that shortens the tape contents. Every cycle ends with a restart operation that forces the transducer to reset the internal state to the initial one and output some symbols. After a finite number of cycles, it halts and accepts (or rejects) while also outputting some symbols.

The second new model of transductions are *transducing observer systems* introduced by the present authors [8]. They are derived from the paradigm of Computing by Observing, which intends to model the way in which information is gained *via* experiments in natural sciences. Often while the actual experiment is running, the results are produced by repeatedly measuring certain quantities like temperature, population size, etc. This was formalized in the way depicted in Figure 2.

Originally this model was introduced for generating and accepting formal languages [4], but the idea of observing and writing a protocol translates very naturally into transductions. The evolving system, though, is not an experiment but a string-rewriting system in our case. The fact that these evolve in discrete derivation steps and the configurations are just plain strings makes them very suitable for this function.

The main aim of the work presented here is to establish relations between the classes of relations computed by restarting transducers, transducing observer systems, and the classical hierarchy of transductions. To this end we first introduce all three types of devices. Then transducing observer systems with length-reducing

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

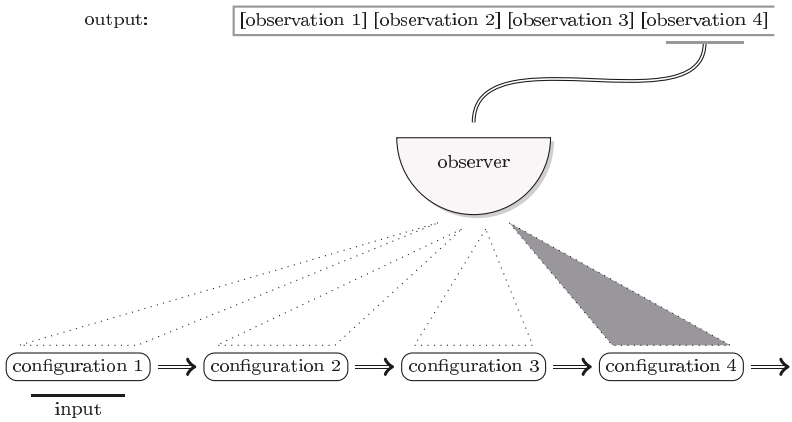


FIGURE 2. Schematic representation of a transducing observer system.

string-rewriting rules form the center of our investigations. In Section 3 we relate them to pushdown transductions, more exactly speaking to pushdown functions; in Section 4 we show that they compute a class of relations very similar to the one computed by RRWW-transducers. Then we shift our attention to simpler string-rewriting systems (*i.e.* painter systems) and finally show that transducing observer systems using this simple type of rules, are more powerful than the mechanisms considered before.

Some of the results presented here have appeared in conference proceedings. More specifically, Sections 3 and 4 are based on a contribution to NCMA 2012 [9] and Section 5 partially on a contribution to NCMA 2010 [8].

2. Definitions and examples

The reader is assumed familiar with standard terminology and notations from Formal Language Theory.

2.1. Transducing observer systems

The observed systems in our architecture will be string-rewriting systems. Concerning these we follow notations and terminology as exposed by Book and Otto [3]. A *string-rewriting system* W on an alphabet Σ is a subset of $\Sigma^* \times \Sigma^*$. Its elements are called rewrite rules, and are written either as ordered pairs (ℓ, r) or as $\ell \to r$ for $\ell, r \in \Sigma^*$.

A string w is called *irreducible* with respect to W, if no rewrite rule from W can be applied to it, *i.e.* it does not contain any factor that is the left hand side of a rule. The set of all such strings is denoted by $\mathsf{IRR}(W)$.

By imposing restrictions on the set of rewriting rules, many special classes of rewriting systems can be defined. Here we are only interested in the following

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

two special types of rewriting systems. A string-rewriting system is called *length-reducing* if for all its rules (ℓ, r) we have $|\ell| > |r|$, that is, every rule shortens the string by at least one symbol. A string-rewriting system is called a *painter* system if for all its rules (ℓ, r), we have $|\ell| = |r| = 1$, that is, every rule just replaces one letter by another one.

In the role of the observer we use a slightly modified version of the device that have become standard in this function: monadic transducers. A *generalized monadic transducer* (gMT for short) is a tuple $\mathcal{O} = (Q, \Sigma, \Delta, \delta, q_0, \phi)$, where the set of states Q, the input alphabet Σ, the transition function δ, and the start state q_0 are the same as for deterministic finite automata. Δ is the output alphabet, and ϕ is the output function, a mapping $Q \to \Delta^*$ which assigns an output word that may also be empty. The class of all generalized monadic transducers is denoted by $g\mathcal{MT}$.

The generalized monadic transducer works as follows. It reads the input word and then produces as output the image under ϕ of the state it stops in.

Now we combine the two components, a string-rewriting system and a monadic transducer in the way described in the introduction.

Definition 2.1. A *transducing observer system*, short T/O system is a triple $\Omega = (\Sigma, W, \mathcal{O})$, where Σ is the input alphabet, W is a string-rewriting system over an alphabet Γ such that $\Sigma \subseteq \Gamma$ which consists of all the symbols that occur in the rule set W, and $\mathcal{O}$ is a generalized monadic transducer, whose input alphabet is Γ.

The mode of operation of a transducing observer system $\Omega = (\Sigma, W, \mathcal{O})$ is as follows: the string-rewriting system starts to work on an input word u. After every rewriting step the observer reads the new string and produces an output. The concatenation of all observations of a terminating derivation forms the output word v. The relation that Ω computes consists of all possible pairs (u, v). Note that already the input string is the first observation; thus there can be an output even if no rewriting rule can be applied to the first string.

Further, the observer is equipped with an important feature: by outputting the special symbol $\perp$ it can abort a computation. In that case no output is produced. The other way in which no output might be produced is, if the string-rewriting system does not terminate. Formally, the relation computed is

$$\mathrm{Rel}(\Omega) = \{(u, v) \mid \exists s : s \in W(u) \text{ and } v = \mathcal{O}(s) \text{ and } |v|_\perp = 0\},$$

where $W(u)$ denotes all sequences of words $(u, u_2, \ldots, u_k)$ that form terminating derivations $u \Rightarrow_W u_2 \Rightarrow_W \ldots \Rightarrow_W u_k$ of W, and $\mathcal{O}(s)$ is $\mathcal{O}(u) \cdot \mathcal{O}(u_2) \ldots \mathcal{O}(u_k)$ for such a sequence. Thus $\mathrm{Rel}(\Omega)$ consists of all pairs of input words combined with the observations of possible terminating derivations on the given input word. Last but not least the class of relations defined by a special type of transducing observing system is denoted by $\mathcal{R}\mathrm{el}$.

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

Example 2.2. Let $\Omega = (\Sigma, W, \mathcal{O})$ be the lr-T/O-system with $\Sigma = \{a\}$, $W = \{aa \to A, AA \to B, BB \to C\}$ and the observer $\mathcal{O}$ that is defined as:

$$\mathcal{O}(w) = \begin{cases} \varepsilon & w \in (a^8)^* \\ a^2 & w \in A^+a^* \\ b^4 & w \in B^+A^* \\ c^8 & w \in C^+B^* \\ \perp & \text{else.} \end{cases}.$$

Starting from any word from a^*, only the rule $aa \to A$ can be applied until all a have been consumed from left to right. Every other rule would lead to a string containing B at the same time as a; the observer would output $\perp$ and thus invalidate the transduction. For every two a that are deleted, also two a are output.

In the same way, all A are reduced to B, then all B to C. Every time the number of symbols is divided by two. So for one C there must be two B, four A and eight a as predecessors. During the computation, eight copies of each a, b and c are produced. If the original number of a is not divisible by eight, then the observer rejects the input.

Thus, the relation computed by the transducing observer system is

$$R = \mathrm{Rel}(\Omega) = \{(a^n, a^n b^n c^n) \mid n \equiv 0 (\mathsf{mod}\ 8)\}.$$

2.2. Restarting transducers

A restarting transducer (RRWW-Td for short) is defined as a 9-tuple $T = (Q, \Sigma, \Delta, \Gamma, ¢, \$, q_0, k, \delta)$ where Q is the finite set of states, Σ and Γ are the finite input and tape alphabet, Δ is the finite output alphabet, $¢, \$ \notin \Gamma$ are the markers for the left and right border of the tape, $q_0 \in Q$ is the initial state and $k \geq 1$ is the size of the read/write window. The transition function δ is of the form

$$\delta : Q \times \mathcal{PC}^{(k)} \to \mathfrak{P}_{fin}(Q \times (\{\mathsf{MVR}\} \cup \mathcal{PC}^{\leq (k-1)}) \cup \{\mathsf{Restart}, \mathsf{Accept}\} \times \Delta^*),$$

where $\mathcal{PC}^{(k)}$ denotes the set of possible contents (over Γ) of the read/write window of T and $\mathfrak{P}_{fin}$ denotes the set of all finite subsets of its argument. The transducer works in cycles, where each cycle is a combination of a number of move-right-steps, one rewrite-step and a restart step. If a restarting transducer enters a cycle where no further restart operation is performed, we call this the tail of a computation. Hence, such a tail computation (and thus the whole computation) ends with an accept-step or by simply getting stuck. Additionally note that in case of a tail computation the rewrite-step is optional. Every rewrite step of the form $(q, v) \in \delta(p, u)$ shortens the tape, that is $|u| > |v|$. After a rewrite step is applied, the read/write window is placed immediately to the right of the string v. Further, the output of the transducer, that is a word in Δ^*, is produced during a restart-step at every end of a cycle or during an accept-step in a tail computation. In the following we explain these steps in more detail. For this we use the notion of configurations.

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

A configuration of T is described by a pair $(\alpha q\beta, z)$, where $\alpha q\beta$ ($\alpha\beta \in \mathfrak{c} \cdot \Gamma^* \cdot \$$, $q \in Q$) is a configuration of the underlying restarting automaton and $z \in \Delta^*$ is the output produced so far. The three most important situations that may occur during a computation of the transducer T are described by the following configurations. At the beginning of a computation T is in the initial configuration $(q_0\mathfrak{c}w\$, \varepsilon)$, where $w \in \Sigma^*$ is the input word and the output is empty. The restarting configuration is described by $(q_0\mathfrak{c}w'\$, v)$, where $w' \in \Gamma^*$ is a subword of w possibly annotated with some auxiliary symbols and $v \in \Delta^*$ is the current output. And (Accept, z) denotes the accepting configuration, where $z \in \Delta^*$ is the output word of T.

Accordingly an accepting computation of T consists of a finite sequence of cycles that is followed by an accepting tail computation. It can be described as

$$\begin{aligned}(q_0\mathfrak{c}w\$, \varepsilon) \vdash_T^c (q_0\mathfrak{c}w_1\$, v_1) \vdash_T^c \ldots \vdash_T^c (q_0\mathfrak{c}w_m\$, v_1 \ldots v_m)\\ \vdash_T^* (\mathsf{Accept}, v_1 \ldots v_m v_{m+1}),\end{aligned}$$

where $w_1, \ldots, w_n \in \Gamma^*$ and $v_1, \ldots, v_{m+1} \in \Delta^*$. Notice again that the operations that can be performed within every cycle are only a certain number of move-right steps and one rewrite step.

Obviously $\vdash_T$ denotes the single step relation and $\vdash_T^c$ denotes the execution of a complete cycle. Finally $\vdash_T^*$ and $\vdash_T^{c^*}$ are the reflexive and transitive closures of these relations.

Now from a static point of view, we associate a binary relation with T, that is,

$$\mathrm{Rel}(T) = \{(w, z) \in \Sigma^* \times \Delta^* \mid (q_0\mathfrak{c}w\$, \varepsilon) \vdash_T^* (\mathsf{Accept}, z)\}.$$

More dynamically T defines a transduction such that every input word $w \in \Sigma^*$ is mapped onto the set of words $z \in \Delta^*$ for which there exists a path in the graph of $\mathrm{Rel}(T)$. Thus $T(w) = \{z \in \Delta^* \mid (w, z) \in \mathrm{Rel}(T)\}$. Now the image of a language $L \subseteq \Sigma^*$ produced by T is defined as $T(L) = \bigcup_{w \in L} T(w)$ and the preimage is $T^{-1}(L) = \{u \in \Sigma^* \mid T(u) \cap L \neq \emptyset\}$. Last but not least the class of relations defined by one type of restarting transducer is denoted by $\mathcal{R}\mathrm{el}$.

Further on we want to introduce the notion of meta-instructions for restarting transducers. Meta-instructions were used to increase the readability of the behavior of restarting automata (*e.g.* exposed in [14]). For that we briefly recall that a tuple of the form $(E_1, u \to u', E_2)$ mirrors the cycle of an RRWW-automaton that reads across the tape content E_1, rewrites a subword u by a shorter subword u' and finally the automaton checks if the part of the tape unseen until then corresponds to E_2. As these meta-instructions describe the rewriting behavior of an automaton they can easily be extended to restarting transducers. Now

$$(E_1, u \to u', E_2; v)$$

is a restarting transducer's meta-instruction, where E_1, E_2, u, u' are defined as for the corresponding automaton and v is the output word produced at the end

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

of this cycle. To describe accepting tail computations we use meta-instructions of the form $(E, \mathsf{Accept}; v)$.

Finally a restarting transducer described as RWW-Td denotes a RRWW-Td that must immediately restart after performing a rewrite instruction.

Example 2.3. Let $T = (Q, \{a, b, c\}, \{\hat{a}, \hat{b}, \hat{c}\}, \{a, b, c, B, C\}, ¢, \$, q_0, 3, \delta)$ be the RRWW-Td that is described by the following meta-instructions:

$$\begin{array}{llll} (1)\ (¢ \cdot (abc)^*, & abc \to Bc, & (Bc)^* \cdot \$; & \hat{a}), \\ (2)\ (¢ \cdot (Bc)^*, & Bc \to C, & C^* \cdot \$; & \hat{b}), \\ (3)\ (¢ \cdot C^*, & C \to \varepsilon, & \$; & \hat{c}), \\ (4)\ (¢ \cdot \$, & \mathsf{Accept}; & & \varepsilon). \end{array}$$

Obviously T consumes only words from the input language $L = \{(abc)^n | n \geq 0\}$. Thus, at the beginning of the computation it scans the tape from left to right while checking the correct order of a's, b's and c's. Next T deletes stepwise, from right to left, all a's and produces the same number of $\hat{a}$'s. Then the transducer proceeds to do the same for b's and c's. Observe that the behavior of T when producing $\hat{a}$'s is nondeterministic, as it must guess the rightmost unrewritten subword abc in every cycle. In order to do so it is clear that $T(L) = \{\hat{a}^n \hat{b}^n \hat{c}^n \mid n \geq 0\}$ and the graph of that transduction is $\mathrm{Rel}(T) = \{((abc)^n, \hat{a}^n \hat{b}^n \hat{c}^n) | n \geq 0\}$. From the meta-instructions it is clear how the transition function can be designed.

2.3. Rational and pushdown relations

Finally, we present in short two classical classes of relations. First of all, the *rational (word) relations* (RAT for short) are defined as the rational subsets of the product of two free monoids. The probably best known characterization of this class is in terms of finite state transducers (FST for short), which are nondeterministic finite automata (with ε-steps), where an output word is assigned to every step of the automaton.

Secondly, in parallel to the characterization of context-free languages by pushdown automata or context-free grammars, there are two similar approaches for transductions. The first one is in terms of pushdown transducers (PDT for short). According to definition of a finite state transducer a PDT is derived from the definition of a pushdown automaton by assigning an output word to every step of the automaton. $\mathcal{R}\mathrm{el}(\mathsf{PDT})$ is the class of relations computed by pushdown transducers; they are also called the *pushdown relations* PDR.

Details on both previously defined relation classes can be found for example in the overview by Choffrut and Culik [6].

The second characterization of the pushdown relations is made by the so called simple syntax directed translation scheme (sSDTS for short). Roughly speaking a sSDTS is a combination of two context-free grammars that are controlled both by the same nonterminals. Thus a sSDTS is defined as 5-tuple $D = (V, \Sigma, \Delta, P, S)$, where V is a finite set of nonterminals, Σ is a finite input- and Δ a finite output alphabet, P is a finite set of rules of the form $A \to \alpha, \beta$, where $\alpha \in (V \cup \Sigma)^*$,

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

$\beta \in (V \cup \Delta)^*$ and the nonterminals in β occur in the same order as in α. Hence, each rule in P can be described as $A \to x_1 B_1 x_2 B_2 \dots x_n B_n,\ y_1 B_1 y_2 B_2 \dots y_n B_n$ for $x_i \in \Sigma^*$, $y_i \in \Delta^*$, and $B_i \in V \cup \{\varepsilon\}$ $(1 \leq i \leq n)$. This property is actually the reason why the defined syntax directed translation scheme is called simple. Finally S denotes the start symbol. The relation generated by D is defined as $\mathrm{Rel}(D) = \{(u, v) \mid (S, S) \Rightarrow_D^* (u, v)\}$. Here we do not get into more detail but recommend Aho and Ullman's book as a reference [2]. There also the following equivalences are shown: $\mathcal{R}\mathrm{el}(\mathsf{sSDTS}) = \mathcal{R}\mathrm{el}(\mathsf{PDT}) = \mathsf{PDR}$.

We conclude this subsection by introducing some important proberties of the relation classes introduced before. In general, we call a relation R *length-bounded*, if and only if there is a constant c, such that for each pair $(u, v) \in R$ with $u \neq \varepsilon$, $|v| \leq c \cdot |u|$ holds. Accordingly, the class of all *length-bounded pushdown relations* is denoted by lbPDR.

Further, we call a relation function if it is *single valued*, that is, for every input word u there is at most one output word v. Then, it is clear that the class of single valued relations generated by sSDTS coincides with the class of pushdown functions (PDF for short).

Additionally, the following results taken from [1], which refer to the length-bounded property, are of importance.

Proposition 2.4. *If $R \in \mathcal{R}\mathrm{el}(\mathsf{sSDTS})$, then there is a constant c, such that for all $u \neq \varepsilon$ in the domain of R there is a v, such that (u, v) is in R and $|v| \leq c \cdot |u|$.*

Corollary 2.5. *Let $R \in \mathcal{R}\mathrm{el}(\mathsf{sSDTS})$ be a single valued relation. Then there is a constant c, such that if $u \neq \varepsilon$ and (u, v) is in R then $|v| \leq c \cdot |u|$.*

Thus, every single valued relation generated by a simple syntax directed translation scheme is length bounded. It follows that this is also true for the class of pushdown functions. Furthemore, it is an immediate consequence that PDF is properly included in lbPDR.

3. Length-reducing systems

We start our investigations by focusing on the relations that are computed by observer systems, where the underlying string rewriting system only uses length-reducing rules ($\mathsf{lr\text{-}T/O}$ for short). Note that a each relation defined by such a system is *length-bounded* in the sense of the definition in the latter section. This is based on the simple observation that in every step the length of the input string has to be decreased, while only a finite number of symbols can be produced. For these systems we obtain the following consequence of Example 2.2.

Proposition 3.1. *The class of pushdown relations is incomparable to the class of relations computed by $\mathsf{lr\text{-}T/O}$-systems.*

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

Proof. Clearly, PDR contains relations that do not fulfill the property that the length of the output word is somehow bounded by the length of the input word. For example, the relation $R = \{(\varepsilon, c^n) \mid n \in \mathbb{N}\}$, is easily computed by pushdown transducer, but it is not computable by any lr-T/O-system.

On the other hand, Example 2.2 provides a lr-T/O-system that computes the relation $R = \{(a^n, a^n b^n c^n) \mid n \equiv 0(\mathsf{mod}\ 8)\}$. This relation is derived by "accepting" the regular language a^n $(n \equiv 0(\mathsf{mod}\ 8))$ and outputting the context-sensitive language $a^n b^n c^n$. It is well known that pushdown transducers are only capable to map regular languages onto context-free languages (see [7]). □

When we restrict our attention from the class of pushdown relations to its length-bounded subclass, we obtain an inclusion up to the empty input. Here we use pushdown transducers with the same restrictions that Jancar et al. used on pushdown automata which they simulated by restarting automata [13]. These restrictions are useful for us, too, since restarting automata work in a way similar to transducing observer systems.

Theorem 3.2. *Each length-bounded pushdown relation $R \subseteq \Sigma^* \times \Delta^*$ that contains only one pair in the form of (ε, v) (with $v \in \Delta^*$) is computable by a* lr-T/O*-system.*

Proof. Proposition 3.1 shows that the inclusion is proper. It remains to show that for any relation $R \in$ lbPDR that respects the Theorem's restriction, there is a lr-T/O-system Ω such that $R = \mathrm{Rel}(\Omega)$. We first prove that we can impose the following restrictions on the pushdown transducers:

(1) the pushdown transducer is nondeterministic and accepts by an empty pushdown;
(2) before it increases the height of its pushdown by one symbol, it must read at least two input symbols;
(3) it reads at least one symbol in every step.

In terms of languages it is easy to verify that there is a pushdown automaton for any context-free language that fulfills these three conditions. This special machine is derived by using a grammar in Greibach normal form and standard compression techniques to control the height of the pushdown store.

As it is not straightforward to verify that there is a pushdown transducer of the above type for any length-bounded pushdown relation, we first explain how a transducer of such a special form can be obtained.

Aho and Ullman [1] showed that every pushdown relation that is defined by a sSDTS can also be defined by a sSDTS D in quadratic Greibach normal form, that is, all rules are of the form $A \to (a\alpha, b\alpha)$, where a is a symbol of the input alphabet or the empty word, b is a symbol of the output alphabet or the empty word, a and b are not both the empty word and α is a string of nonterminals of length at most two.

Now let us assume that $D = (V, \Sigma, \Delta, P, S)$ is a sSDTS in quadratic Greibach normal form that generates a length-bounded relation. Thus, we can make some

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

additional assumptions on the form of the rules of D. That is, any derivation that produces non-empty output on empty input has to be of bounded length. More formally, there is a positive integer k such that for any derivation of the form $(A, A) \Rightarrow_D^* (X_A, yX_A)$, where $A \in V$, $X_A \in V^*$ and $y \in \Delta^*$, the length of the output string y is bounded by k (and therefore also the number of nonterminals $|X_A|$), that is $|y| \leq k$. The latter statement holds for the reason that if there is no such integer k, this would violate the property of being length-bounded. From this it is clear that for every $k \in \mathbb{N}$ there exists a sSDTS $D_1 = (V, \Sigma, \Delta, P_1, S)$ that is derived from D by eliminating rules of the form $A \to (\alpha, b\alpha)$. Thus P_1 contains all rules of P with the following exceptions:

- If $A \to (\alpha, b\alpha)$ is in P, then it is not in P_1, where $\alpha \in V \cup V^2$ and $b \in \Delta$.
- If there is a left most derivation of the form $(A, A) \Rightarrow_D^{k' \leq k} (X_A, yX_A) \Rightarrow_D (aBX'_A, ybBX'_A)$, then the rule $A \to (aBX'_A, ybBX'_A)$ is in P_1, where $A \in V$, $a \in \Sigma$, $b \in \Delta$, y is an output word, B is a nonterminal or empty and finally X'_A is a nonempty string of nonterminals[3].

Thus, in a strict sense D_1 is not a syntax directed translation scheme, as here possible strings of output symbols are produced during one derivation step. Further on D_1 is not necessarily in quadratic Greibach normal form. Nonetheless a proof that $\mathrm{Rel}(D) = \mathrm{Rel}(D_1)$ is straightforward if done by induction on the length of a derivation.

Continuing, let M_1 be a pushdown transducer for D_1, similar to the one exposed in *e.g.* [2], that simulates left derivations of D_1 in its pushdown store. Thus, if M_1's topmost pushdown symbol is an A and the sSDTS D_1 has a rule of the form $A \to (a\alpha, y\alpha)$ where $\alpha \in V^*$ and $y \in \Delta^*$, then M_1 replaces A by α, checks whether the current input symbol is an a and outputs y. Recall that $\alpha \in V^*$, a is a single input symbol and y is an output word or the empty word. Notice that M_1 already fulfills the conditions (1) and (3) from above and from this it is clear that in any accepting computation the height of its pushdown store in step i is at most $k \cdot i$.

From here on it is easy to see that M_1 can be transformed into a pushdown transducer M which uses shorter pushdown store, that is, in step i the height of the pushdown store is at most $\frac{i}{2} + 1$. This can be done by a standard compression technique, where one pushdown symbol of M encodes $2k$ pushdown symbols of M_1. The machine M derived in this way also fulfills the condition (2) from above. Additionally notice that a pushdown transducer of this form will in general not exist for a pushdown relation that is not length-bounded, as the height of the pushdown stack will not necessarily be bounded.

Up to now we have seen that for any length-bounded pushdown relation there is a pushdown transducer M of the special form described. Now M can easily be simulated by a lr-T/O-system $\Omega = (\Sigma, W, \mathcal{O})$. Here the basic idea is that the current state of M and the content of the pushdown store is encoded in the input string of Ω left of the current input symbol of M. Without loss of generality

[3]Clearly the construction of D_1 depends on the knowledge of k.

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

the pushdown alphabet is disjoint from the input alphabet. Let $t_1 : (q_1, a, A) \to (q_2, A, v_1)$ and $t_2 : (q_2, b, A) \to (q_3, BC, v_2)$ be two consecutive transitions of M, where q_i is a state, a, b are input symbols, A, B, C are pushdown symbols and v_1, v_2 is the current output. So this transition can be applied in a configuration, where the head of the pushdown transducer is over a letter a in state q_1, and the top of the pushdown store is A. In our representation this corresponds to a string $U[A, q_1, v]abx$, where U is a string of pushdown symbols, v is the output associated to the previous step and x is a string of input symbols. Then these two steps of the pushdown transducer correspond to one derivation step of our string rewriting system W, that is:

$$U[A, q_1, v]abx \Rightarrow UC[B, q_3, v_1v_2]x.$$

Note that the state is represented in a compound symbol with the last pushdown symbol A next to the current input a.

After this we are back in a configuration equivalent to the one before we started the simulation. Thus, the next transition can be simulated. The length-reducing rules necessary for the simulation are obvious from the derivation. Further, it is clear that a derivation of the above form is only possible if and only if there are corresponding transitions for the pushdown transducer, because all the corresponding symbols must be present in the required positions. The observer is constructed in a way that it executes the following tasks: it has to verify the correct occurrence of pushdown and input symbols within the given string and outputs the output string v_1v_2 given in the compound of the last pushdown symbol. Thus, a generalized monadic transducer $\mathcal{O}$ can control the correct simulation of the transitions by admitting strings of the forms described and by rejecting the computation, if any other type of string appears.

Finally, if the length-bounded pushdown relation contains only one pair (ε, v) we set $\mathcal{O}(\varepsilon) = v$. Observe, that we are not able to handle more than one pair of the latter form for the reason that the observer is defined deterministic. □

Next we state an immediate consequence of the latter result and the fact that PDF $\subsetneq$ lbPDR. Here observe that every pushdown function necessarily contains at most one pair in the form of (ε, v).

Corollary 3.3. PDF $\subsetneq$ Rel(lr-T/O)

4. The relation to restarting transducers

Now we try to relate $\mathcal{R}$el(lr-T/O) to the classes of relations computed by restarting transducers. We cannot establish an exact equivalence, but we will see that RRWW-transducers compute very similar relations.

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

Proposition 4.1. $\mathcal{R}\mathrm{el}(\mathsf{lr\text{-}T/O}) \subseteq \mathcal{R}\mathrm{el}(\mathsf{RRWW\text{-}Td})$

Proof. Let $\Omega = (\Sigma, W, \mathcal{O})$ be a $\mathsf{lr\text{-}T/O}$ system with $\mathcal{O} = (Q, \Gamma, \Delta, \delta, q_0, \phi)$ and the string rewriting system W is defined as a finite set over $\Gamma^* \times \Gamma^*$. From Ω we build an RRWW-transducer $T = (Q', \Sigma, \Delta, \Gamma', ¢, \$, q_0, k, \delta')$ where k is at least as large as the longest left hand side of a rule $l \to r$ in W and $\Gamma' = \Gamma \cup \{\bar{a} \mid a \in \Gamma\}$.

The main idea of the simulation is that T combines the application of a rule and the observer's behavior. In every cycle T scans the whole tape and determines the output of $\mathcal{O}$ for the current input, that is actually the result of an application of a rule of W to the tape content of the previous cycle, which belongs to the description of $\mathcal{O}$. Meanwhile, T also nondeterministically applies a rule from W somewhere, to shorten the tape content. It is easy to verify that a restarting automaton for the latter behavior is fully described by meta-instructions of the form $(¢ \cdot \Gamma^*, l \mapsto r, \Gamma^* \cdot \$)$ for every rule $l \to r$ in W. Further on, adding the observer to these instructions by applying a standard technique for the intersection of two automata we derive the intended transducer T. Obviously the output of T, when seeing the \$-symbol corresponds to the output function of $\mathcal{O}$, that is $(\mathsf{Restart}, v) \in \delta'(q', \$)$ if and only if $\phi(q) = v$. Here q' denotes a state of T that corresponds to q of the observer. Note that in case $\phi(q) = \perp$, T is simply designed such that it gets stuck, that is, there is $\delta'(q', \$) = \emptyset$.

Finally we have to verify accepting conditions for T. Notice that in terms of transducing observer systems, acceptance means that no rule of the string rewriting system is applicable, that is, the current string belongs to $\mathsf{IRR}(W)$. It is well known that $\mathsf{IRR}(W)$ is a regular language. Hence, the set of meta-instructions of T described above simply has to be extended such that also the regular language $\mathsf{IRR}(W)$ is taken into account. Then T recognizes if the current tape content belongs to $\mathsf{IRR}(W)$. In this case it accepts while seeing the \$-symbol and it outputs the corresponding symbols that $\mathcal{O}$ has produced.

The simulation of the transducing observer system Ω by the RRWW-transducer T is quite direct. Thus it is straightforward to see that the same input/output pairs are produced. □

We strongly suspect that the inverse of Proposition 4.1 does not hold. This would mean that there are relations that can be computed by RRWW-transducers but not by length-reducing $\mathsf{T/O}$-systems. The latter cannot directly connect the output to the rule that has been applied. Typically an intermediate step is used to indicate to the observer, which rule has been applied where. This trace must then be deleted. If all rules are shortening, only about every second one can be used to produce output in this way, while an RRWW-transducer can rewrite and output (even more than one symbol) in every step. However, we can prove a weakened variant of the inverse inclusion of the one in Proposition 4.1.

Proposition 4.2. *For every relation $R \subseteq \Sigma^* \times \Delta^*$ and $R \in \mathcal{R}\mathrm{el}(\mathsf{RRWW\text{-}Td})$ that contains only one pair in the form of (ε, v) $(v \in \Delta^*)$, there is a uniform morphism φ and a relation $S \in \mathcal{R}\mathrm{el}(\mathsf{lr\text{-}T/O})$ such that $R = \{(u, v) \mid (\varphi(u), v) \in S\}$.*

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

Proof. The basic idea of this proof is to simulate the rewriting steps of a restarting transducer by the rules of a length-reducing system and the prefix (and also suffix) parts, which belong to a regular language, by the observer. For that, the morphism φ transforms words into a redundant representation with a copy of each letter. This allows us to do more than one step in the deletion of a letter by deleting also the copy. The latter is needed to "clean up" after applying a rule. In this way, the lr-T/O-system can simulate the RRWW-transducer in a very direct way.

Let $T = (Q, \Sigma, \Delta, \Gamma, ¢, \$, q_0, k, \delta)$ be an RRWW-transducer. The morphism is defined as $\varphi(x) = x\hat{x}$ for all $x \in \Gamma$, where the $\hat{x}$ is a copy of the original symbol. Thus, every letter is mapped into two copies. The string-rewriting rules that the T/O-system Ω uses are derived from the meta-instructions of T. Let

$$t : (E_1, u \cdot x \to u', E_2; v)$$

be such a transition for $u, u' \in \Gamma^*$ and $x \in \Gamma$. We associate to each transition a unique label, here t. From this description we build a lr-T/O-system $\Omega = (\Sigma, W, \mathcal{O})$ as follows, where $W \subseteq \Gamma' \times \Gamma'$ and Γ' consists of $\Gamma \cup \{\hat{x} \mid x \in \Gamma\}$ as well as some special symbols described below. The string-rewriting rule, which is added to W, for the meta-instruction t is $\varphi(ux) \to \varphi(u') \cdot t$. Thus, the additional space induced by φ is used to assign the corresponding label to the applied instruction.

Further we add $t \to \varepsilon$. While $x\hat{x}$ is deleted, the observer produces the string that T outputs in the execution of t. As described in the meta-instruction, this string is v. For every meta-instruction of T the observer's mapping includes the clause

$$\mathcal{O}(w) = v; \text{ if } w \in \varphi(E_1) \cdot \varphi(u') \cdot t \cdot \varphi(E_2).$$

In this clause, we check whether t was really applicable. This means that T can reach the state in which the rewrite operation of t is applied after reading the prefix of the current string until the application site of the rule. Of course, the part ux is not there anymore, when this clause is applied. But since there is the symbol t, a monadic transducer can recognize this symbol and act as if ux was there, that is, the current string must belong to the language $\varphi(E_1) \cdot \varphi(u') \cdot t \cdot \varphi(E_2)$ rather than to $E_1 \cdot u \cdot E_2$, which are the strings that allow application of the meta-transition t. This is still a regular condition to check. Note that all the clauses, which describe the observer, are disjoint, due to the presence of t. Thus, the observer always produces the desired output. Additionally after applying a meta-instruction the label t has to be deleted by the rule $t \to \varepsilon$ and in this case the observer has a clause of the form

$$\mathcal{O}(w) = \varepsilon; \text{ if } w \in \varphi(\Gamma^*).$$

We treat accepting cycles in an analogous way. In the case that a computation of T accepts with empty tape such that the observer system has also no symbols left, then clearly, Ω stops as well, with the same output as T. Note that according to our restriction, a relation defined by a restarting transducer that accepts with empty tape obviously contains the pair (ε, v), where $v \in \Delta^*$. Clearly, here the observer has a clause $\mathcal{O}(\varepsilon) = v$. The second case is more problematic, where T accepts and

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

Ω has still symbols left. As long as there are symbols left, we can rewrite any of them to a symbol of an accepting transition t_a as above.

Observe that Ω will not stop when the symbol t_a is introduced. But the computation of a T/O-system is complete only when no more rule can be applied. That is why we use the special symbol t_a to delete every remaining symbol, that is, we add rules $xt_a \to t_a$ and $t_a x \to t_a$ for all symbols x in the alphabet of W. That is, t_a absorbs all the symbols that are left. With the presence of t_a the observer maps any further string to the empty output. So when there is only t_a left, the system stops and has output the same string as T.

Finally, we explain how Ω behaves on rejecting computations of T. Note, that T rejects an input word simply by getting stuck, that is, no transition is applicable in the current configuration. As the clauses of the observer mirror directly the move-right steps of T, it will also get stuck. In this situation we have to output $\perp$ to abort the computation of Ω. This can be done by making the observer "complete", that is, the transition function of $\mathcal{O}$ is extended such that every input word w, which is not described by the regular expressions above leads to the following output: $\mathcal{O}(w) = \perp$. This completes the proof. □

5. Painter systems

Now we turn our attention to a simpler type of string-rewriting system. In *painter systems* for every rule $l \to r$ the length bound $|l| = |r| = 1$ holds. Thus only one symbol is replaced by a different one. Although the rewrite rules are simpler, they have the possibility of making loops of the form $a \Rightarrow b \Rightarrow a$. Compared to length-reducing rules, this gives them more time for computing and this results in higher computational power.

Proposition 5.1. *Every relation that is computable by a* lr-T/O*-system is also computable by a* pnt-T/O*-system.*

Proof. In general, painter rules can usually simulate any kind of rule that does not prolong a string in the context of Computing by Observing. This has been shown for example in the work on acceptors [5] in Theorem 4.1 and the following corollaries. Therefore we only sketch the main idea here.

For this we look at a length-reducing rule $r : ab \to c$. Since one symbol is rewritten and one is deleted, it illustrates the two things that a length-reducing rule can do to a position in a string. This rule will be simulated by the derivation $ab \Rightarrow r_1 b \Rightarrow r_1 r_2 \Rightarrow c r_2 \Rightarrow c\lambda$ by the obvious painter rules. Note that every intermediate string contains at least one of r_1 and r_2; thus their presence indicates that the simulation of rule r is going on. During this phase the observer does not need to produce any output, since these steps do not occur in the length-reducing system. It only watches that no other combination of symbols than the ones in the derivation above occur, and that no other simulation starts. For bigger rules, if more than one symbol is rewritten, all of them are treated as the a above, more deletions are treated like the b.

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

When all the r-symbols are gone, we have the same string as obtained by $ab \to c$ with one exception: the λ symbol is there. It indicates that one symbol has been deleted in this place. This information, however, is irrelevant for the further derivation. Therefore the observer should simply ignore this symbol, *i.e.* read over it without changing the state. Similarly, the observer must allow any derivation $a\lambda^i b \Rightarrow^* c\lambda^{i+1}$ besides the $ab \Rightarrow^* c\lambda$ explained above as long as the rewrite steps are the same; this is done by adding λ^* between all the letters of the clauses of the observer. For example, let Σ be the input alphabet in the example above. Then the observer should map all words from $\Sigma^* r_1 b \Sigma^*$, $\Sigma^* r_1 r_2 \Sigma^*$, and $\Sigma^* c r_2 \Sigma^*$ to the empty string. λ from previous steps are accommodated by changing these clauses to $(\{\lambda\} \cup \Sigma)^* r_1 \lambda^* b (\{\lambda\} \cup \Sigma)^*$, $(\{\lambda\} \cup \Sigma)^* r_1 \lambda^* r_2 (\{\lambda\} \cup \Sigma)^*$, and $(\{\lambda\} \cup \Sigma)^* * c\lambda^* r_2 (\{\lambda\} \cup \Sigma)^*$ respectively.

Then the observer can produce the same string of outputs as the observer of the original length-reducing system. The sequence of observations is not exactly the same, because there are several empty observations during the simulation of longer length-reducing rules. □

When we try to relate T/O-systems with painter rules to restarting transducers we run into the same problem as exposed in the preceding section. On empty input the observer is not able to produce several outputs. Thus, we restrict our results in the same way as before.

Proposition 5.2. *Every relation $R \subseteq \Sigma^* \times \Delta^*$ and $R \in \mathcal{R}\text{el}(\text{RRWW-Td})$ that contains only one pair in the form of (ε, v) ($v \in \Delta^*$), is computable by a* pnt-T/O-*system.*

Proof. Recall that we have stated a similar result in Proposition 4.2. There we applied a uniform morphism on the input to gain some space. This additional space was needed to save the trace of which meta-instruction was applied. Here, as the rules of the current observer system are not length-reducing, all tape cells can be rewritten any number of times. Hence, there is no need for additional space. Thus, we can save the trace directly together with the current input string.

Without loss of generality, let $T = (Q, \Sigma, \Delta, \Gamma, ¢, \$, q_0, k, \delta)$ be a proper RRWW-transducer that restarts and accepts only on the \$-symbol. Further, assume that T is described by labeled meta-instructions, where each rewriting meta-instructions of T is defined as follows:

$$t : (E_1, u_1 u_2 \dots u_k \to u'_1 u'_2 \dots u'_k, E_2; v)\,,$$

where $u_i \in \Gamma$ and $u'_i \in \Gamma \cup \{\varepsilon\}$ $(i \in \{1, \dots, k\})$ is the corresponding symbol that occurs in the string produced by the current meta-instruction. Note that we here use a slightly different representation of meta-instructions, that is, each rewriting rule $u \to u'$ is described as a unique letter to letter mapping, where at least one letter is mapped to the empty word. For instance, let $u = u_1 u_2 \dots u_k$ and $u' = u'_1 u'_2 \dots u'_k$, where $u_i, u'_i \in \Gamma \cup \{\varepsilon\}$ $(i \in \{1, \dots, k\})$, then the rule $u \to u'$ can be depicted as

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

$u_1u_2\dots u_k \to u'_1u'_2\dots u'_k$, which implies that u_1 is rewritten by u'_1 and so on. Further, each accepting meta-instruction is also uniquely labeled by $t_a : (E, \mathsf{Accept}; \varepsilon)$. From the description of T we build a pnt-T/O-system $\Omega = (\Sigma, W, \mathcal{O})$, where the observer is defined as $\mathcal{O} = (Q', \Gamma' \cup \{\text{¢}, \$\}, \Delta, \delta', p_0, \phi)$ and we consider the string rewriting system W is a set of rules over $(\Gamma' \cup \{\text{¢}, \$\})^* \times (\Gamma' \cup \{\text{¢}, \$\})^*$, where Γ' consists of Γ, the set $\{a^t, a^{tt}, a^{t_a} \mid a \in \Gamma \cup \{\text{¢}, \$\}$ and t, t_a are labels of rewriting or accepting meta-instructions of $T\}$, the special auxiliary symbols λ and f, which denote erasing rules, and the set $\{\lambda^a, a^\lambda \mid a \in \Gamma \cup \{\$\}\}$, which is needed to "remove" erased symbols. In the following, the latter set is needed to simulate accepting meta-instructions by the observer system. Additionally we assume that every input string of W is also bounded by the markers ¢ and \$. At the end of the proof we will show that this is not a necessary assumption. Anyway, it increases the readability of the technical details.

Here we show for one concrete case how the set of rewriting rules W is defined, and how the observer is used to control the derivation of W.

For each meta-instruction $t : (E_1, u_1u_2\dots u_k \to u'_1u'_2\dots u'_k, E_2; v)$ of T, we add the rules $u_i \to u_i^t$, $u_i^t \to u_i^{tt}$ and $u_i^{tt} \to u'_i$ $(i \in \{1, \dots, n\})$ to W. If $u'_i = \varepsilon$, then we add the rule $u_i^{tt} \to \lambda$, where λ is an auxiliary symbol. Now the observer's mapping for an input $w \in (\Gamma \cup \{\text{¢}, \$\})^*$ includes the clauses

$$\mathcal{O}(w) = \begin{cases} \varepsilon; & \text{if } w \in E_1 \cdot u_1u_2\dots u_k \cdot E_2, \\ \varepsilon; & \text{if } w \in E_1 \cdot u_1^t u_2\dots u_k \cdot E_2, \\ \vdots & \vdots \\ \varepsilon; & \text{if } w \in E_1 \cdot u_1^t u_2^t \dots u_k^t \cdot E_2, \\ \varepsilon; & \text{if } w \in E_1 \cdot u_1^{tt} u_2^t \dots u_k^t \cdot E_2, \\ \vdots & \vdots \\ v; & \text{if } w \in E_1 \cdot u_1^{tt} u_2^{tt} \dots u_k^{tt} \cdot E_2, \\ \varepsilon; & \text{if } w \in E_1 \cdot u_1^{'} u_2^{tt} \dots u_k^{tt} \cdot E_2, \\ \vdots & \vdots \\ \varepsilon; & \text{if } w \in E_1 \cdot u_1^{'} u_2^{'} \dots u_k^{'} \cdot E_2. \end{cases}$$

Let $t : (E_1, u_1u_2\dots u_i\dots u_k \to u'_1u'_2\dots u'_i\dots u'_k, E_2; v)$ be a meta-instruction that is applicable to a restarting configuration $(q_0\text{¢}xu_1u_2\dots u_ky\$, v')$, where $\text{¢} \cdot x \in E_1$, $y \cdot \$ \in E_2$, $u'_i = \varepsilon$, and $v' \in \Delta^*$. Hence, this leads to a cycle of the form:

$$(q_0\text{¢}xu_1u_2\dots u_ky\$, v') \vdash_T^c (q_0\text{¢}xu'_1u'_2\dots u'_ky\$, v'v).$$

Then the transitions of T are simulated by Ω as follows:

$$\begin{array}{rl} & \text{¢}xu_1u_2\dots u_i\dots u_ky\$ \Rightarrow \text{¢}xu_1^tu_2\dots u_i\dots u_ky\$ \Rightarrow \\ \cdots \Rightarrow & \text{¢}xu_1^tu_2^t\dots u_i^t\dots u_k^ty\$ \Rightarrow \text{¢}xu_1^{tt}u_2^t\dots u_i^t\dots u_k^ty\$ \Rightarrow \\ \cdots \Rightarrow & \text{¢}xu_1^{tt}u_2^{tt}\dots u_i^{tt}\dots u_k^{tt}y\$ \Rightarrow \text{¢}xu'_1u_2^{tt}\dots u_i^{tt}\dots u_k^{tt}y\$ \Rightarrow \\ \cdots \Rightarrow & \text{¢}xu'_1u'_2\dots\lambda\dots u'_ky\$ \end{array}.$$

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

After this we are back in a situation similar to the one before we started the simulation. The only difference is that after the first simulation of a meta-instruction, the configuration of W is interspersed with the λ-symbol. These special symbols have to be "removed", as otherwise possible rewriting rules of W, which are taken from the description of the restarting transducer, might not be applicable. Recall that a painter system is not allowed to delete any symbol. Hence, we need to introduce additional rules for W and clauses for the observer such that every occurring λ is shifted to the right of the \$-symbol before the next rule of T is applied. This can be done by applying a technique exposed in [5] to simulate context-sensitive rules in the form of $AB \to BA$ by painter systems. Hence, without going into details we can obviously add rules to W such that the observer enables the following derivation

$$\lambda a \Rightarrow_W \lambda^a a \Rightarrow_W \lambda^a a^\lambda \Rightarrow_W a a^\lambda \Rightarrow_W a\lambda,$$

for every $a \in \Gamma \cup \{\$\}$. Clearly, every clause added to the observer to apply the previous derivation leads to an empty output. Further, the special λ-symbols right of the \$-symbol are ignored; that is, it reads over them without a change of state. Thus, the next transition can be simulated. Further notice, that all the intermediate configurations can be described by disjoint regular expressions that are specific to the transition t, or the symbols λ^a, a^λ, because they contain some versions of these symbols.

By exhaustive checking, we can verify that application of these rules in any other order, or the application of a rule stemming from another transition will lead to a string not described by these expressions. Therefore, it is shown that a generalized monadic transducer can control the correct simulation of these transitions by admitting strings of the forms described and by rejecting the computation, if any other type of string appears.

Now for each accepting meta-instruction $t_a : (E, \mathsf{Accept}; v)$, we add the rules $a \to a^{t_a}$ and $a^{t_a} \to f$ to W, where $a \in \Gamma \cup \{\text{¢}, \$\}$. Then the observer only has to check whether t_a occurs in its input and whether the input corresponds to the regular language E. The special symbol f is again needed to force the system to stop, that is, when f is introduced, there are no rules to "delete" f. Thus, after f occurs in a configuration, there possibly is a finite number of additional rewrite steps. Hence, the observer is defined such that all configurations where the symbol f is included lead to the empty output. Observe that there is only one meta-instruction of the above form, where $E = \text{¢}\varepsilon\$$. In this case the we set $\mathcal{O}(\varepsilon) = v$.

Finally, by adding additional symbols and rewriting rules to Ω that mark the first and last letter of the input string at the beginning of the computation, we can omit the assumption that every input string contains the special symbols ¢ and \$. Clearly, then every rule in W and the accepting conditions have to be adjusted So it is shown that for every RRWW-transducer T, there is a pnt-T/O-system Ω such that $\mathrm{Rel}(T) = \mathrm{Rel}(\Omega)$. □

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

Note one difference in the proofs of Propositions 5.1 and 5.2: whereas in the first case we leave the λ in place and make the observer slightly more complicated, in the second we move the λ to the right. Either method would work in either case here.

The inclusion in Proposition 5.2 is proper, as it is clear that the relation $\{(a, a^n) \mid n \geq 1\}$ is computable by the pnt-T/O-system $\Omega = (\Sigma, W, \mathcal{O})$, where $W = \{a \to a, a \to f\}$ and

$$\mathcal{O}(w) = \begin{cases} a; & \text{if } w = a, \\ \varepsilon; & \text{if } w = f, \\ \bot; & \text{if } w \neq a \text{ and } w \neq f. \end{cases}$$

Note that this particular relation violates the length-bounded property. This implies that also the inclusion $\mathcal{R}\text{el}(\text{lr-T/O}) \subsetneq \mathcal{R}\text{el}(\text{pnt-T/O})$ is proper.

The example of $\{(a, a^n) \mid n \geq 1\}$ shows that relations computed by painter systems are not necessarily length-bounded. In contrast to restarting transducers we can show that nearly every rational relation is computable by such a system.

Theorem 5.3. *Each rational relation $R \subseteq \Sigma^* \times \Delta^*$ that contains only one pair in the form of (ε, v) ($v \in \Delta^*$) is computable by a* pnt-T/O*-system.*

Proof. This result seems quite obvious. Nevertheless, it is not an immediate consequence of the results presented in this paper. By Proposition 5.1 and by the fact that lr-T/O-systems compute the length-bounded pushdown relations (*cf.* Thm. 3) up to empty input it is clear that there is a pnt-T/O-system for every relation $R \in \text{lbPDR}$, where there is only on pair $(\varepsilon, v) \in R$ with v is a word over the output alphabet. Further, as lbPDR is obviously a superclass of the rational relations for which the length-bounded property holds, we only have to show how a pnt-T/O system works on rational relations that are not length-bounded. Note that the violation of this property is caused by finite state transducers that produce nonempty output, when performing cycles without reading any symbol.

To begin with, we may assume that $\Omega = (\Sigma, W, \mathcal{O})$ is a pnt-T/O-system that simulates a finite state transducer $T = (Q, \Sigma, \Delta, \delta, q_0, F)$ that computes a length-bounded rational relations. We further assume that Ω is designed similar to the lr-T/O system that simulates a pushdown transducer shown in the proof of Theorem 3. Thus, when T performs a step $(qabu, v) \vdash_T (pbu, v\alpha)$ with a transition in the form of $t : \delta(q, a) = (p, \alpha)$, where $q, p \in Q$, $a, b \in \Sigma$, $u \in \Sigma^*$ and $v, \alpha \in \Delta^*$, then Ω simulates this step by the following derivation,

$$\begin{aligned} \lambda^*[q, a]bu &\Rightarrow_W \lambda^*[q, a, p, \alpha]_t bu \quad &&\Rightarrow_W \lambda^*[q, a, p, \alpha]_t b^t u \\ &\Rightarrow_W \lambda^*[q, a, p, \alpha]_t[p, b]u &&\Rightarrow_W \lambda^*\lambda[p, b]u, \end{aligned}$$

where λ, $[q, a]$, $[q, a, p, \alpha]_t$, b^t and $[p, b]$ are auxiliary symbols not in Σ. Admittedly, this representation might be quite redundant, but it is clear from the derivation and the results cited above, how the rules W are obtained from the transitions of T. Further, the different configurations in a derivation can obviously be described by disjoint regular expressions, which is the definition of the observer $\mathcal{O}$. Additionally,

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

the observer outputs α when scanning the configuration $\lambda^*[q, a, p, \alpha]_t bu$. Up to now this simulation is simply a consequence of previous results, as stated above.

We next describe how Ω mirror possible ε-steps of a finite state transducer, which causes the violation of the length-bounded property. For that we add a transition in the form of $t_\varepsilon : \delta(q, \varepsilon) = (q', \beta)$ to the transition function of T, where q is the state used above, $q' \in Q$ and $\beta \in \Delta^*$. Thus, we have to extend the string rewriting systems by rules that are in some sense derived by calculating the ε-closure[4] of a state of T. Here, for q we have to add to the rule $[q, a] \to [q, a, p, \alpha]_t$ a rule $[q, a] \to [q, a, q', \beta]_{t_\varepsilon}$ and $[q, a, q', \beta]_{t_\varepsilon} \to [q', a]$, which mirrors the returning to the actual configuration of the finite state transducer. Clearly by introducing the special symbol t_ε the observer can be adjusted such that it enables derivations of W in the form of

$$\lambda^*[q, a]bu \Rightarrow_W \lambda^*[q, a, q', \beta]_{t_\varepsilon} bu \Rightarrow_W \lambda^*[q', a]bu,$$

and that it outputs β when reading $\lambda^*[q, a, q', \beta]_{t_\varepsilon} bu$. Further, observe that the example transition t_ε already covers all cases of ε-transitions occurring in a description of a finite state transducer. Finally, if the rational relation R contains one pair (ε, v) $(v \in \Delta^*)$ we simply add the clause $\mathcal{O}(\varepsilon) = v$ to the observer. For that note, a finite state transducer that computes such a relation R is not able to perform a cycle of ε-steps on empty input, as this would violate the property that only one pair is in the form of (ε, v). Hence, v can easily be obtained by calculating a kind of ε-closure for the initial state q_0 of T. Thus, it is shown that for every relation $R \in \mathsf{RAT}$, which is restricted in the above way, there is a pnt-T/O-system Ω such that $R = \mathrm{Rel}(\Omega)$. □

This result cannot be extended to compute all the pushdown relations by pnt-T/O-systems.

Proposition 5.4. *The pushdown relations are incomparable to* $\mathcal{R}\mathrm{el}(\mathsf{pnt\text{-}T/O})$.

Proof. Clearly, by Proposition 5.2, there are relations computed by pnt-T/O that are not pushdown relations. Conversely, besides the trivial example of producing several outputs on the empty input, consider the relation $R = \{(c, a^n b^n) \mid n \geq 0\}$. Obviously R is a pushdown relation. A pushdown transducer for R uses ε-steps to push a number of symbols on the stack while outputting the same number of a's. At some point of the computation it decides nondeterministically to pop all stack symbols while again outputting the same number of b's. Finally it just has to check that there is only one c on the tape.

A pnt-T/O-system for R gets c as input string. Hence, it must rewrite c to produce an output. As there is only a finite number of rewriting rules, c has to be rewritten in a form of cycle-rules to produce an arbitrary number of a's. Thus, it is obvious that there is no possibility to save the number of rules that were used to

[4]The ε-closure of a state q is the set of states that are reachable from q by zero or more ε-transitions (*e.g.* in [15], p. 52).

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

produce a's. Therefore, the pnt-T/O-system is not able to output the same number of b's. Hence, $R \notin \mathcal{R}\text{el}(\text{pnt-T/O})$. □

6. Perspectives

One of our main aims in introducing the class of length-reducing transducing observer systems was the following: they seemed to be a good candidate for a class of transductions between the ones defined by RRWW- and RWW-transducers. RWW-Td have to restart immediately after a rewrite step and thus cannot read the remainder of the tape. It is an open problem whether these two classes are equal. This is a variant of the corresponding question for restarting automata. For them it is a long-standing open problem whether the possibility of reading on after a rewrite step increases their computational power, see for example the work of Jancar et al. [12].

The results of Section 4 suggest that $\mathcal{R}\text{el}(\text{lr-T/O})$ and $\mathcal{R}\text{el}(\text{RRWW-Td})$ are very similar but not equal. Establishing the relations between $\mathcal{R}\text{el}(\text{lr-T/O})$ and $\mathcal{R}\text{el}(\text{RWW-Td})$ could thus resolve the question of equality between $\mathcal{R}\text{el}(\text{RWW-Td})$ and $\mathcal{R}\text{el}(\text{RRWW-Td})$.

References

[1] A.V. Aho and J.D. Ullman, Properties of Syntax Directed Translations. *J. Comput. Syst. Sci.* **3** (1969) 319–334.
[2] A.V. Aho and J.D. Ullman, The theory of parsing, translation, and compiling. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1972).
[3] R.V. Book and F. Otto, String-rewriting systems. *Texts and monographs in computer science*. Springer (1993).
[4] M. Cavaliere and P. Leupold, Evolution and Observation – A Non-Standard Way to Generate Formal Languages. *Theoret. Comput. Sci.* **321** (2004) 233–248.
[5] M. Cavaliere and P. Leupold, Observation of String-Rewriting Systems. *Fundam. Inform.* **74** (2006) 447–462.
[6] C. Choffrut and K. Culik II, Properties of Finite and Pushdown Transducers. *SIAM J. Comput.* **12** (1983) 300–315.
[7] S. Ginsburg and G.F. Rose, Preservation of Languages by Transducers. *Inf. Control* **9** (1966) 153–176.
[8] N. Hundeshagen and P. Leupold, Transducing by Observing, in vol. 263 of *NCMA*, edited by H. Bordihn, R. Freund, M. Holzer, T. Hinze, M. Kutrib and F. Otto. *books@ocg.at*, Austrian Computer Society (2010) 85–98.
[9] N. Hundeshagen and P. Leupold, Transducing by Observing and Restarting Transducers, in vol. 29 of *NCMA*, edited by R. Freund, M. Holzer, B. Truthe and U. Ultes-Nitsche., *books@ocg.at*, Österreichische Computer Gesellschaft (2012) 93–106.
[10] N. Hundeshagen and F. Otto, Characterizing the Rational Functions by Restarting Transducers, in *LATA*, vol. 7183 of *Lect. Notes in Comput. Sci.*, edited by A.H. Dediu and C. Martín-Vide. Springer (2012) 325–336.
[11] P. Jancar, F. Mráz, M. Plátek and J. Vogel, Restarting Automata, in *FCT*, vol. 965 of *Lect. Notes in Comput. Sci.*, edited by H. Reichel. Springer (1995) 283–292.

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press

[12] P. Jancar, F. Mráz, M. Plátek and J. Vogel, Different Types of Monotonicity for Restarting Automata, in *FSTTCS*, vol. 1530 of *Lect. Notes in Comput. Sci.*, edited by V. Arvind and R. Ramanujam. Springer (1998) 343–354.
[13] P. Jancar, F. Mráz, M. Plátek and J. Vogel, On Monotonic Automata with a Restart Operation. *J. Automata, Languages and Combinatorics* **4** (1999) 287–312.
[14] F. Otto, Restarting Automata. in vol. 25 of *Recent Advances in Formal Languages and Applications*, edited by Z. Ésik, C. Martín-Vide, and V. Mitrana. Springer (2006) 269–303.
[15] G. Rozenberg and A. Salomaa, Handbook of formal languages, word, language, grammar (vol. 1). Springer-Verlag New York, Inc., New York, USA (1997).

Communicated by M. Holzer.
Received January 31, 2013. Accepted January 8, 2014.

https://doi.org/10.1051/ita/2014002 Published online by Cambridge University Press